Improved tardiness bounds for Global EDF

Jeremy Erickson   Sanjoy Baruah   UmaMaheswari Devi

University of North Carolina at Chapel Hill

IBM Research Lab, Bangalore, India

September 22, 2010

Task Model

- Sporadic task model

Task Model

- Sporadic task model
  - Every task has a worst-case execution time and minimum separation time

Task Model

- Sporadic task model
  - Every task has a worst-case execution time and minimum separation time
  - Every deadline assumed to be equal to minimum separation time (implicit deadline)

Task Model

- Sporadic task model
  - Every task has a worst-case execution time and minimum separation time
  - Every deadline assumed to be equal to minimum separation time (implicit deadline)
- All tasks independent

Task Model

- Sporadic task model
  - Every task has a worst-case execution time and minimum separation time
  - Every deadline assumed to be equal to minimum separation time (implicit deadline)
- All tasks independent
- Fully preemptible

Task Model

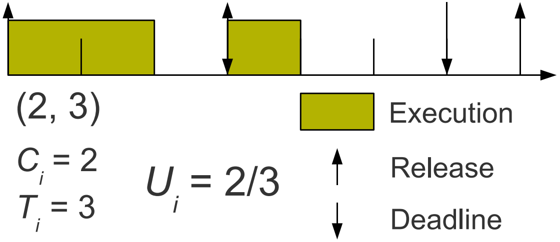
- Sporadic task model
  - Every task has a worst-case execution time and minimum separation time
  - Every deadline assumed to be equal to minimum separation time (implicit deadline)
- All tasks independent
- Fully preemptible
- No self-suspensions

Notation

- m = # of processors

Notation

- m = # of processors
- C_i = Worst-case execution time



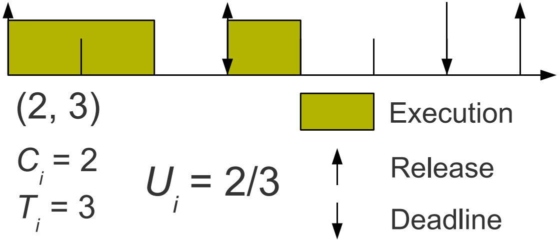
$C_i = 2$

$T_i = 3$

$U_i = 2/3$

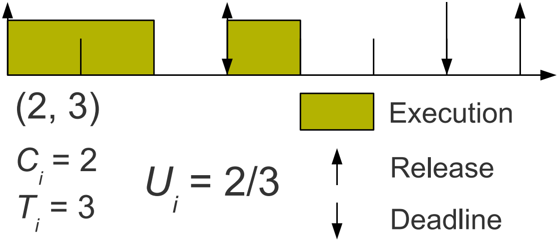
Notation

- m = # of processors
- C_i = Worst-case execution time
- T_i = Minimum separation time

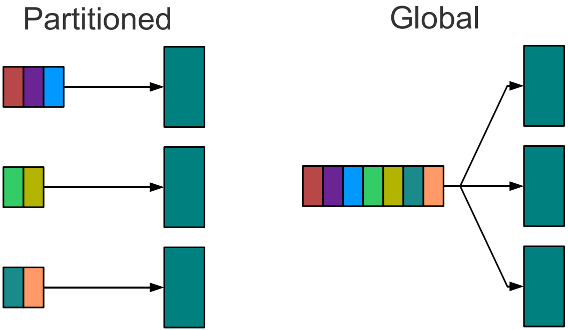


Notation

- m = # of processors
- C_i = Worst-case execution time
- T_i = Minimum separation time
- U_i = A task's *utilization* C_i / T_i



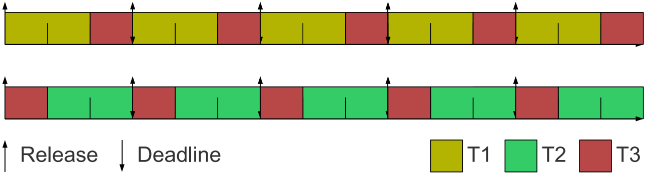
Scheduler (Global EDF)



- **EDF** = Earliest Deadline First
- Here we consider the behavior of global EDF

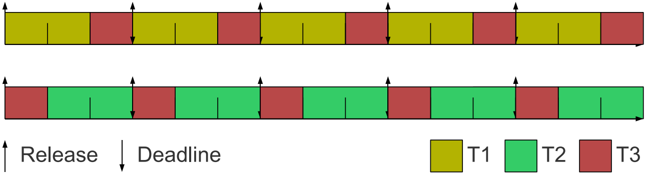
Hard Real-time

- **Hard Real-time** = all deadlines met



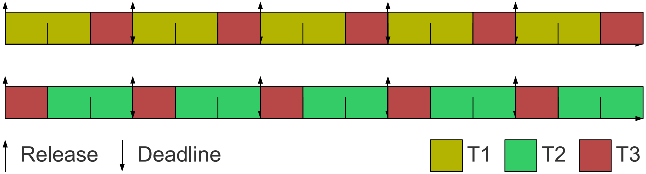
Hard Real-time

- **Hard Real-time** = all deadlines met
- Schedulable if:
  - $\forall i, U_i \leq 1$ and
  - $\sum U_i \leq m$



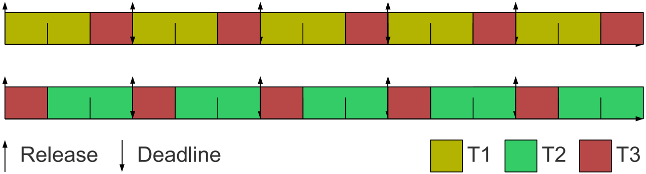
Hard Real-time

- **Hard Real-time** = all deadlines met
- Schedulable if:
  - $\forall i, U_i \leq 1$ and
  - $\sum U_i \leq m$
- Requires context switch time to be accounted for in U_i



Hard Real-time

- **Hard Real-time** = all deadlines met
- Schedulable if:
  - $\forall i, U_i \leq 1$ and
  - $\sum U_i \leq m$
- Requires context switch time to be accounted for in U_i
- Number of context switches may be huge!



Soft Real-time

- **Soft Real-time** = bounded tardiness

Soft Real-time

- **Soft Real-time** = bounded tardiness
- Sufficient for broad range of applications

Soft Real-time

- **Soft Real-time** = bounded tardiness
- Sufficient for broad range of applications
- Schedulable under same conditions as HRT, **but** may reduce total context switch cost and thus U_i values

Soft Real-time

- **Soft Real-time** = bounded tardiness
- Sufficient for broad range of applications
- Schedulable under same conditions as HRT, **but** may reduce total context switch cost and thus U_i values
  - Global EDF provides SRT schedulability with many fewer context switches than algorithms such as PFAIR

Devi/Anderson Bounds - Basic Idea

- Devi & Anderson (2005) provide a method to compute tardiness bounds for global EDF.

Devi/Anderson Bounds - Basic Idea

- Devi & Anderson (2005) provide a method to compute tardiness bounds for global EDF.
- Bound tardiness of each task at $x + C_i$ for some x.

Devi/Anderson Bounds - Basic Idea

- Devi & Anderson (2005) provide a method to compute tardiness bounds for global EDF.
- Bound tardiness of each task at $x + C_i$ for some x.
- Nontrivial part is finding x.

Devi/Anderson Bounds - Basic Idea

- Devi & Anderson (2005) provide a method to compute tardiness bounds for global EDF.
- Bound tardiness of each task at $x + C_i$ for some x.
- Nontrivial part is finding x.
- Bound does vary per task, but x does not.

Devi/Anderson Bounds - Basic Bound

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of global EDF.

Devi/Anderson Bounds - Basic Bound

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of global EDF.
- Derived in 2005 conference paper (“Naive Bound”):

m - 1 largest C_i values

smallest C_i value

$$x = \frac{C_{\text{sum}} - C_{\text{min}}}{m - U_{\text{sum}}}$$

CPU count

m - 1 largest U_i values

Devi/Anderson Bounds - Prior Improvements

- Devi & Anderson 2005 also presents improved bounds.
- EDF-BASIC: Use only $m-2$ utilization values.
- Further improved EDF-ITER: Like EDF-BASIC, but only use values from selected $m-1$ tasks.

Our Improvement

- We present improvements that apply to both the naive and EDF-ITER techniques.

Our Improvement

- We present improvements that apply to both the naive and EDF-ITER techniques.
- Use different x_i value for each task.

Our Improvement

- We present improvements that apply to both the naive and EDF-ITER techniques.
- Use different x_i value for each task.
- Thus, we deal with a vector $\vec{x}$ instead of a single x.

Our Improvement

- We present improvements that apply to both the naive and EDF-ITER techniques.
- Use different x_i value for each task.
- Thus, we deal with a vector $\vec{x}$ instead of a single x.
- In worst case, becomes same results as Devi/Anderson.

Our Improvement

- We present improvements that apply to both the naive and EDF-ITER techniques.
- Use different x_i value for each task.
- Thus, we deal with a vector $\vec{x}$ instead of a single x.
- In worst case, becomes same results as Devi/Anderson.
- Only a summary of resulting differences given here.

$\mathbf{L}(\vec{x})$

- Define a function $\mathbf{L}(\vec{x})$ used while evaluating whether a proposed $\vec{x}$ produces valid bounds.

$\mathbf{L}(\vec{x})$

- Define a function $\mathbf{L}(\vec{x})$ used while evaluating whether a proposed $\vec{x}$ produces valid bounds.
-
$$\mathbf{L}(\vec{x}) = \sum_{(m-1)\ \text{largest}} \left(x_i U_i + C_i \right) \tag{1}$$
  - Improves on naive bound in Devi/Anderson

$\mathbf{L}(\vec{x})$

- Define a function $\mathbf{L}(\vec{x})$ used while evaluating whether a proposed $\vec{x}$ produces valid bounds.
-
$$\mathbf{L}(\vec{x}) = \sum_{(m-1)\ \text{largest}} \left(x_i U_i + C_i\right) \tag{1}$$
    - Improves on naive bound in Devi/Anderson
- Can use improved definition $\mathbf{L}(\vec{x})$: the largest sum obtained by summing $(m-2)$ of the $(x_i U_i + C_i)$'s plus an additional C_i.
    - Improves on EDF-ITER in Devi/Anderson

Compliant Vector

- Using $\mathbf{L}(\vec{x})$ as defined, a vector is *compliant* iff $\forall i$,

$$\frac{\mathbf{L}(\vec{x}) - C_i}{m} \leq x_i \tag{2}$$

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ denote any compliant vector. For each task τ_i, each job generated by τ_i completes no later than $(C_i + x_i)$ time units after its deadline.

- Proof is fundamentally similar to that of Devi and Anderson, although with notational differences.

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ denote any compliant vector. For each task τ_i, each job generated by τ_i completes no later than $(C_i + x_i)$ time units after its deadline.

- Proof is fundamentally similar to that of Devi and Anderson, although with notational differences.
- By utilizing x_i instead of x, we can bound tardiness of a specific task under consideration more tightly.

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ denote any compliant vector. For each task τ_i, each job generated by τ_i completes no later than $(C_i + x_i)$ time units after its deadline.

- Proof is fundamentally similar to that of Devi and Anderson, although with notational differences.
- By utilizing x_i instead of x, we can bound tardiness of a specific task under consideration more tightly.
- This allows the proof to pull through using the definition of "compliant vector" above.

Theorem 1 - Proof Details

- Rather than using **LAG** (as in previous papers), use $W(t)$

Theorem 1 - Proof Details

- Rather than using **LAG** (as in previous papers), use $W(t)$
- I = set of jobs with deadlines no later than t.

Theorem 1 - Proof Details

- Rather than using **LAG** (as in previous papers), use $W(t)$
- I = set of jobs with deadlines no later than t.
- $W(t) = \sum_{\text{jobs in } I}(C_i - \text{work completed before } t)$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$
  - Not tardy at end, but running - contribute $U_j(d_k - t_{i+1}) + C_j$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$
  - Not tardy at end, but running - contribute $U_j(d_k - t_{i+1}) + C_j$
- Summing contributions reveals claimed upper bound

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k
- Bound follows from here

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k
- Bound follows from here
- After this, we're done

Minimal Compliant Vector

- In light of the theorem, we would like to find a “small” compliant vector

Minimal Compliant Vector

- In light of the theorem, we would like to find a “small” compliant vector
- We define a compliant vector as *minimal* if reducing any one component would produce a non-compliant vector.

Minimal Compliant Vector

- In light of the theorem, we would like to find a “small” compliant vector
- We define a compliant vector as *minimal* if reducing any one component would produce a non-compliant vector.
- Now how do we compute it?

Algorithm for computing minimal compliant vector

FINDCOMPLIANTVECTOR

$\vec{x} \leftarrow \langle 0, 0, \ldots, 0 \rangle$ ▷ Initialize (to a non-compliant vector)
repeat
    Let τ_i denote any task violating constraint
    Let $\hat{x}_i$ denote smallest value of x_i satisfying constraint
    Replace x_i by $\hat{x}_i$ in $\vec{x}$
until $\vec{x}$ is a compliant vector

Minimality of Computed Vector

Theorem

Procedure FINDCOMPLIANTVECTOR *returns a minimal compliant vector.*

Lemma

For all $j \geq 0$, $\mathbf{L}(\vec{x}_j) \leq \mathbf{L}(\vec{x}_f)$.

- Increasing an x_i value can only increase $\mathbf{L}(\vec{x})$.

Minimality of Computed Vector

Theorem

Procedure FINDCOMPLIANTVECTOR *returns a minimal compliant vector.*

Lemma

For all $j \geq 0$, $\mathbf{L}(\vec{x}_j) \leq \mathbf{L}(\vec{x}_f)$.

- Increasing an x_i value can only increase $\mathbf{L}(\vec{x})$.
- Each bound, when set, was tight, so at end, all bounds tight.

Complexity

- No bound known on runtime - seems very large from experiments

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ
- Runs tens to thousands of iterations with $\epsilon = .1$ in experiments

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ
- Runs tens to thousands of iterations with $\epsilon = .1$ in experiments
- Additive error bounded by $m\epsilon$

Experimental Setup

- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$

Experimental Setup

- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$
- Generated random sets of tasks with 1,000 sets for each experiment

Experimental Setup

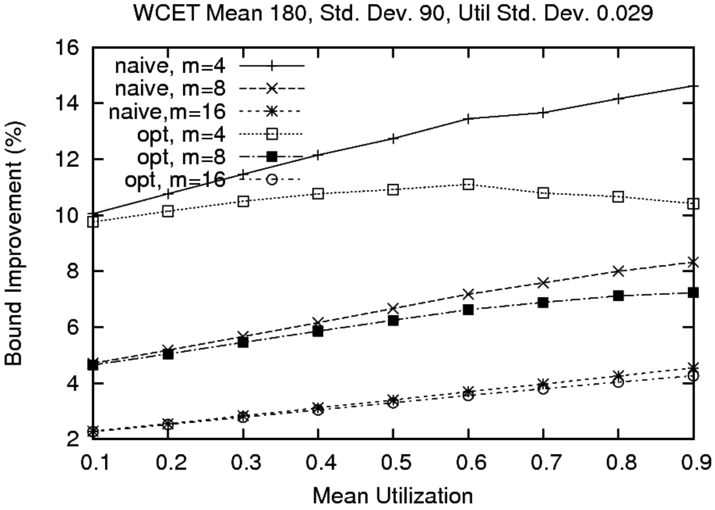
- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$
- Generated random sets of tasks with 1,000 sets for each experiment
- Randomly selected WCET and utilization for each task

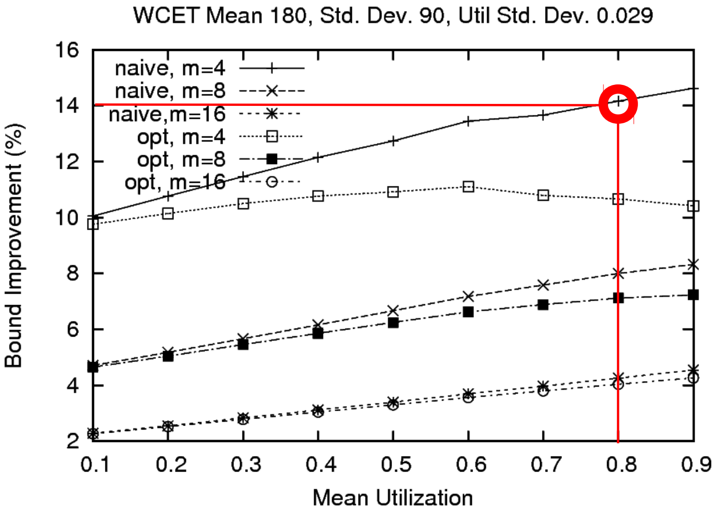
Experimental Setup

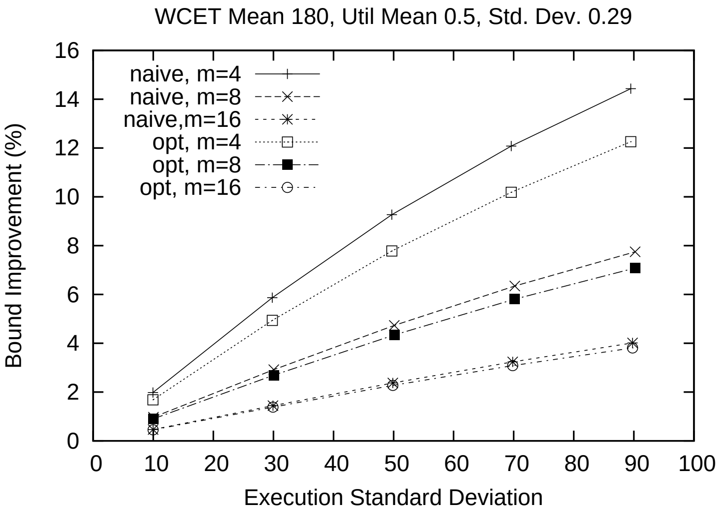
- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$
- Generated random sets of tasks with 1,000 sets for each experiment
- Randomly selected WCET and utilization for each task
- Always used uniform distribution over some interval

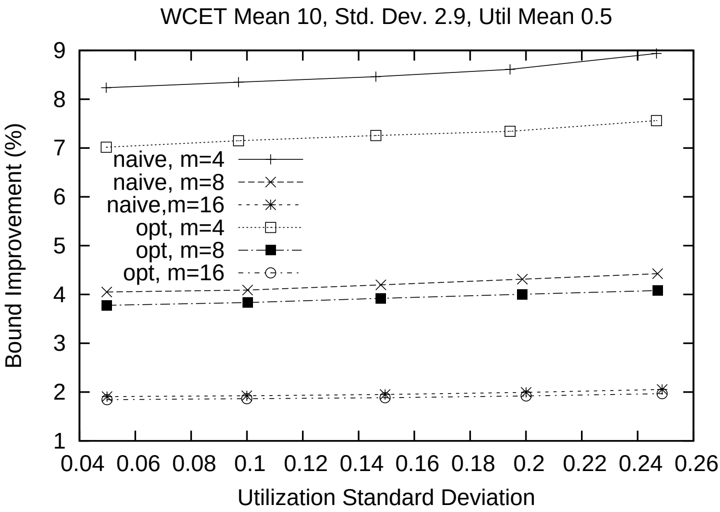
Experimental Setup

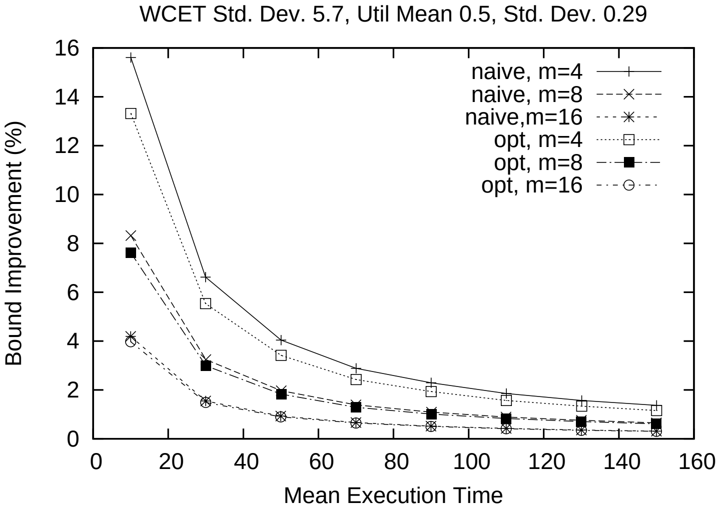
- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$
- Generated random sets of tasks with 1,000 sets for each experiment
- Randomly selected WCET and utilization for each task
- Always used uniform distribution over some interval
- Experiments tested differing mean and variance of WCET and utilization, as well as differing number of CPUs


WCET Mean 180, Std. Dev. 90, Util Std. Dev. 0.029
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
Bound Improvement (%)
Mean Utilization
16
14
12
10
8
6
4
2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


WCET Mean 180, Std. Dev. 90, Util Std. Dev. 0.029
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
Bound Improvement (%)
Mean Utilization
16
14
12
10
8
6
4
2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


WCET Mean 180, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement (%)
Execution Standard Deviation
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
0
2
4
6
8
10
12
14
16
0
10
20
30
40
50
60
70
80
90
100


WCET Mean 10, Std. Dev. 2.9, Util Mean 0.5
Bound Improvement (%)
9
8
7
6
5
4
3
2
1
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0.22
0.24
0.26
Utilization Standard Deviation
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16


WCET Std. Dev. 5.7, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement (%)
Mean Execution Time
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
0
2
4
6
8
10
12
14
16
0
20
40
60
80
100
120
140
160

Review

- Provided optimized bounds for global EDF schedule by using multiple x_i values.
- Evaluated bounds experimentally

Thank You!